university of groningen / faculty of science and engineering

On Conceptor Control in Continuous Time

Bachelor's Thesis

Daniël Woonings, s3992497, d.a.woonings@student.rug.nl,
Supervisors: S. Abreu, MSc & G. A. Pourcel MSc

Abstract: Conceptors have been a successful mechanism used in a variety of tasks ranging from machine learning to neuroscience. As of now, they have always been used in discrete-time contexts. This report extends the theory of conceptors to continuous time objects such as the leaky-integrator echo state network and liquid state machine. In particular, the pattern regeneration task is solved for the continuous-time leaky integrator echo state networks using two different methods. The first method is derived from the assumption of a hard conceptor and is based on orthogonal vector projection. The second method is derived for arbitrary conceptors and the negation of a conceptor is used to construct a vector projection. As a consequence of the first method, several properties of hard conceptors are derived and proved. The derived theory is illustrated by two experiments, the first studies pattern regeneration in continuous time. The second experiment uses the conceptor classifier mechanism in combination with a spiking reservoir. Finally, pattern regeneration for spiking reservoirs is discussed

1 Introduction

The goal of this report is twofold, the first part aims to develop and study the theory of conceptors in a continuous-time setting. The second goal of the report is to unify the results from (Jaeger, 2014) and (Maass et al., 2002), integrating conceptors with liquid state machines. Conceptors (Jaeger, 2014) are tools to control the internal dynamics of discrete-time echo state networks (Jaeger, 2001). We extend the theory to models that deal with continuous time. All in all, we will see that the pattern regeneration problem (in this report called the control task) for both model paradigms is stated equivalently, while it will be solved differently. Inherent differences in model structure lead to different methods to obtain control. Liquid state machines are often seen as the spiking neuron equivalent to ESNs and are continuous time objects. To put the study of conceptors into a more 'biologically plausible' setting, we aim to integrate conceptors and LSMs. First, we shortly discuss why we investigate conceptors in continuous time with possibly spiking neurons.

1.1 Motivation

Arguably, the most important reason to study conceptors is their ability to represent 'symbols' and mediate between neural and symbolic dynamics. This ability makes conceptors interesting in the field of machine learning, using them as classifiers (Qian & Zhang, 2018), learning mechanism (He & Jaeger, 2018), and tools for explainable AI (Jaeger, 2014). However, many of these applications do not require conceptors to be continuous time objects. We now discuss two distinct challenges where conceptors in continuous time arise. The first is contained in the field of cognitive science, the second arises in the study of computation.

Two cornerstones of cognitive science question how cognition arises and how to simulate intelligent behaviour. The simulation of intelligent behaviour is realised using a cognitive architecture, a description of how cognition arises at a certain level of abstraction. The two most common abstraction levels yield symbolic and neural-dynamic architectures. The problem of integrating these two different modelling paradigms remains an unsolved problem. Cognition studied at the symbolic level has been effective at studying high-level cognition such as reasoning, skill acquisition and attention (Anderson, 2007), (Kieras & Meyer, 1997). Theories and ideas developed at the symbolic level abstract away from neural dynamics and low-level cognition, they cannot explain how low-level cognition is integrated within the symbolic paradigm. On the other hand, studying cognition at the neuron level has also been successful, but at a different set of tasks. Cognitive models that explicitly incorporate some form of neural dynamics are known to be more successful at explaining low-level cognitive phenomena, namely, object recognition (Redmon & Farhadi, 2016) and memory recall mechanisms (Hopfield, 1982). They are useful for modelling this set of tasks but fail to scale to higher cognitive functions.

Conceptors have been proposed as a possible mediator between the symbolic and neural dynamical-level since they arise naturally from neural dynamics and can be seen as symbolic representations. Additionally, several sets of operations have been defined on them, such as Boolean logic. These operations can be interpreted at both the neuron and symbolic levels. Conceptors thus provide a method to mediate between the neuron and symbolic level that is interpretable at both levels. It is therefore of interest to study conceptors in a biologically plausible setting. Nearly all biologically plausible network architectures, such as the Liquid State Machine are modelled in continuous time. Moreover, if one ever wants to simulate constructed architectures on a neuromorphic chip, models need to be able to deal with continuous time. From here we obtain the first motivation of extending conceptors to continuous time.

For some time, the study of 'computation' has been placed into a broader perspective. Unconventional, non-digital computational frameworks and formalisms have been investigated (for a review see Jaeger (2021)) as alternatives to Turing-like computing. Specifically, unconventional computing mechanisms that arise from high-dimensional nonlinear dynamical systems have been of interest. Examples of such modelling paradigms include neuromorphic computing (brain-like computing) and computing using physical phenomena (such as optical computing). Specifically, in neuromorphic computing, conceptors are tools that could aid in understanding how computation can occur in brain-like systems and what the concept of computation entails in this context.

Another reason to study conceptors in continuous-time models is the applications of continuous-time models in machine learning. For instance, leaky-integrator echo state networks provide a way to easily change the time scale of the reservoir, making it possible to learn patterns at different time scales.

1.2 Summary

The structure of this report is as follows; first discrete-time echo state networks are treated after which the theory of conceptors is discussed. To bridge the gap between discrete and continuous dynamics, leaky-integrator echo state networks are discussed. The Liquid State Machine is studied to introduce spiking neurons. The theoretical part of the report is concluded by adapting pattern regeneration to continuous time using continuous-time leaky integrator echo state networks. To illustrate the outlined theory two experiments are performed, to gain more insight into how conceptors act in continuous time and an illustration of the conceptor classification mechanism for spiking networks. Finally, the discussed theory and results are put into a broader perspective and the pattern regeneration problem is discussed for spiking networks.

2 Related Work

2.1 Echo state networks

Echo state networks (Jaeger, 2001) are non-linear dynamical systems with a complete observable state $x(n)$, representing the activation of all neurons at time step n. The network can be presented with an input function $u(n)$, which acts on the state. The central thesis of echo state computing (and reservoir computing in general) is that the state of the network can be understood as an 'echo' function of the input history under some conditions. The activation signal of a neuron is a response signal to input $u(n)$.

Formally, an echo state network is a recurrent neural network consisting of K input units, N internal units and L output units. The activations of the three components are represented by the vectors $u(n) \in \mathbb{R}^K$, $x(n) \in \mathbb{R}^N$ and $y(n) \in \mathbb{R}^L$ respectively. The input units are connected to the internal units by $W^{in} \in \mathbb{R}^{N\times K}$, and the internal units are connected to themselves by $W \in \mathbb{R}^{N\times N}$ and the internal units are connected to the output units by $W^{out} \in \mathbb{R}^{L\times N}$. Finally, each neuron has a bias represented by the vector $b \in \mathbb{R}$ Giving the following dynamics:

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + b) \quad (2.1)$$
$$y(n) = g(W^{out}x(n)) \quad (2.2)$$

2.1.1 Echo state property

The dynamics of Equation (2.1) describe an arbitrary RNN, we now discuss the defining property of the ESN architecture, the echo state property. As stated in the previous paragraph, the state of an ESN can be seen as an echo function of the input history of the network. For ESNs to be useful computational tools the network must satisfy the echo state property (ESP). When the system has the ESP it ensures that the initial condition of the network does not determine its trajectory, meaning that the trajectory of the network is asymptotically independent of its initial condition. For discrete ESNs conditions for the ESP can be algebraically derived (for a review see Yildiz et al. (2012)), they are linked to both W and the driving input u. It can be shown that a discrete-time ESN generally has the ESP if the spectral radius (largest absolute eigenvalue of a matrix) $|\lambda_{max}|$ of the reservoir matrix W is less than 1 (i.e $|\lambda_{max}| < 1$). Note that this is neither a necessary nor a sufficient condition, but rather a general rule of thumb. In all simulations, we want to ensure that our models have the ESP.

2.1.2 Training

Finally, we consider how ESNs can be optimised on supervised learning tasks. Contrary to other recurrent learning schemes, only the readout units W_{out} are adapted in reservoir computing. As we have already stated, feeding the ESN a signal, evokes a response signal for every neuron. The output of the ESN is learnt to be a linear combination of these neural response signals. In later sections, we will see how the readouts are learnt using ridge regression.

2.2 Discrete conceptors

In the previous section, we noted that ESNs construct a representation of an input signal in a high dimensional space and compute output signals by linearly combining the neural response signals. The constructed temporal nonlinear high-dimensional expansion of the input signal is useful for both computation and machine learning. It is in many cases possible to learn the characteristics of these expansions as solutions to supervised machine learning tasks. Conceptors (Jaeger, 2014) build on the idea that expansions constructed by the ESN can be used to symbolise input signals. The guiding idea of conceptors is the fact that when a system is driven by an input signal, its dynamics are approximately contained in a certain region of the state space, meaning the dynamics do not cover the state space entirely. A conceptor is simply a characterisation of this area. Besides being a characterisation of the area occupied by the input signal, they can also be used for controlling the reservoir. Conceptors can map arbitrary states into this characteristic region, which is why they are sometimes referred to as "soft" projection matrices. Additionally, conceptors are sometimes called neural dampeners/filters because when the network is fed $u(n)$, the state is contained in a region of the state-space and conceptors damp/filter the part of the state that is not in this region. We now derive some theory of conceptors in the context of linear algebra. Section (2.2) of this report closely follows (Jaeger, 2014).

2.2.1 Storing signals in the reservoir

One control task for which conceptors are used is to retrieve signals from a reservoir in the absence of a driving input. To do this, a signal must first be stored in the reservoir. This is done by adjusting the reservoir matrix W such that it approximates the input-driven system in the absence of a driving input. Formally, we want to adjust the internal weights such that they are the solution to the following criterion:

$$W = argmin_{\tilde{W}} \sum_{i=1,\dots,j} (W^{*}x(n) + W^{in}u^{j}(n+1) - \tilde{W}x^{j}(n))^2 \quad (2.3)$$

Where W^{*} is the original randomly initialised connection matrix. Finding a solution to Equation (2.3) means the set $\{u^1, u^2, \dots, u^j\}$ of signals are 'loaded' into the reservoir. The reservoir is trained to approximate the state in the absence of driving input $Wx(n+1) \approx W^{*}x(n) + W^{in}u(n+1)$. When we refer to the reservoir as being loaded, we mean that Equation (2.3) has been computed and the adjusted matrix W is used for the reservoir dynamics.

2.2.2 Definition

Suppose we have a discrete-time ESN as in Equation (2.1), loaded with a periodic signal $u(n)$. In this section, our goal is to obtain a matrix $C \in \mathbb{R}^{N\times N}$, such that the system responds as if the input $u(n)$ was fed in, similar to:

$$C\tanh(Wx(n)) \approx \tanh(W^{*}x(n) + W^{in}u(n+1)) \tag{2.4}$$

Suppose we feed the system $u(n)$ for T time-steps, we obtain a set of T, N-dimensional vectors $\{x(0),\, x(1),\, \ldots,\, x(T)\}$. Each vector in this set is an observed state of the network when it is fed the signal. The sampled states can be collected into a so-called state matrix formalised by Definition (2.1), which concatenates all observed state vectors.

Definition 2.1. Given a set of T, N-dimensional state vectors $\{x(0),\, x(1),\, \ldots,\, x(T)\}$ corresponding to input signal $u(t)$. The state matrix is defined as;

$$X = \begin{bmatrix} x(0)\; x(1)\; \ldots\; x(T) \end{bmatrix} \tag{2.5}$$

Ideally, the matrix C should constrain the network dynamics to the subspace spanned by X. Note that X gives a full characterisation of the dimensionality of the network response. The state matrix provides a direct expression for all states that were observed when feeding the system $u(n)$. To characterise the space spanned when driving the discrete-time ESN on an input signal we introduce the following definition.

Definition 2.2 (Characteristic subspace)**.** Given a state-matrix $X \in \mathbb{R}^{N\times T}$ as in Definition (2.1). The characteristic subspace of X is defined as;

$$Im(X) = span\{u_1,\, u_2,\, \ldots,\, u_m\} = \tilde{U} \tag{2.6}$$

When $m \neq N$, the space spanned by the network $\tilde{U}$ does not span all dimensions of the state space. It means that $\tilde{U}$ is a proper linear subspace and the 'area' that the system response covers does not vary in every dimension of the state space. The case when $m \neq N$ is important for later study since we can use tools from linear algebra to study the dynamics in a mathematical setting.

The case when $m = N$, means that the 'area' of the system response varies in all dimensions, i.e the span of $\tilde{U}$ is $\mathbb{R}^N$. Even though the network response might vary in all dimensions, the 'area' does not necessarily cover the entire state space. Therefore, we only want the conceptor to project in the directions of the state space that X actually varies significantly. This is formalised using the PCA of X, or equivalently the SVD of the correlation matrix $R = XX'/T$, where the $'$ denotes the transpose of a matrix. The matrix Σ containing the singular values of $R = U\Sigma U'$, tells us how large the response of the system with respect to the orthonormal column vectors of U. For small singular values, it means that for this dimension the system has nearly no response, meaning that the state is nearly not varied in this direction. To control how much the conceptor projects into a given direction a parameter α called aperture is introduced. The aperture parameter acts on the singular values of the conceptor, we will later discuss the role of aperture in more detail.

As stated previously, given an arbitrary state a conceptor should tell us how the state should evolve when feeding the system the input $u(n)$. So, given a state that is already in $\tilde{U}$, the conceptor should act as the identity map. However, if the state is not in the 'area' covered by X, the conceptor should project the state towards that area. Additionally, we want to control how 'strongly' we project into a given dimension of $\tilde{U}$ using the aperture. Putting these criteria together we define a conceptor in terms of an optimisation problem.

Definition 2.3 (Conceptor)**.** Given a state-matrix as in Definition (2.1) and its corresponding correlation matrix $R = XX'/T$, $\alpha \in (0, \infty)$, the conceptor associated with R and the aperture α is defined by;

$$C(R, \alpha) = argmin_C \mathbb{E}_x[||x - Cx||^2] + \alpha^{-2}||C||^2_{fro} \tag{2.7}$$

Where the fro subscript denotes that the Frobenius norm is used. The first term suggests that C dampens regions of the state space that are not covered by X and the latter that it aims to be the smallest matrix to do so (with respect to the Frobenius norm). It can be shown that $C(R, \alpha)$ can be directly computed from R, as stated in the following theorem:

Theorem 2.1. The conceptor as in Definition (2.7) associated with R and α can be obtained by computing;

$$C(R,\alpha) = R(R + \alpha^{-2}I)^{-1} \tag{2.8}$$

For a proof see (Jaeger, 2014).

2.2.3 Aperture adaptation and hard conceptors

The previous section introduced the aperture of a conceptor arising when the characteristic subspace $\tilde{U}$ spans the entire state space. Abstractly, aperture controls the contraction/expansion of the space to which the conceptor projects. Practically, aperture is a real number $\alpha \in (0,\infty)$ that acts on the singular values of a conceptor. An important property of aperture is that conceptors from the same data source (same state matrix) with different apertures can be recovered from one another. The operation that scales the aperture of a conceptor is called aperture adaptation, it is denoted by $\phi(C,\gamma)$, where C is a conceptor and γ the factor by which the aperture of the conceptor is scaled. Let $C_1(R,\alpha_0)$, $C_2(R,\alpha_1)$ be conceptors as in Definition (2.7), with $0 < \alpha_0$, $\alpha_1 < \infty$, we define a scaling factor $\gamma = \frac{\alpha_1}{\alpha_0}$. To obtain C_2, from C_1 compute:

$$C_2 = \phi(C_1,\gamma) = C_1(C_1 + \gamma^{-2}(I - C_1))^{-1} \tag{2.9}$$

As stated previously, the aperture acts on the singular values of a conceptor. It is possible to derive an algebraic expression for the singular values of an aperture-shifted conceptor. Namely, given a conceptor with singular values s_i and scaling factor γ, scaling the aperture of C by factor γ yields a conceptor with singular values determined by the expression:

$$\begin{cases} s_i/(s_i + \gamma^{-2}(1-s_i))) & 0 < s_i < 1,\ 0 < \gamma < \infty \\ 0 & 0 < s_i < 1,\ \gamma = 0 \\ 1 & 0 < s_i < 1, \gamma = \infty \\ 0 & s_i = 0 \\ 1 & s_i = 1 \end{cases} \tag{2.10}$$

From Equation (2.10) we define the notion of a hard conceptor and state two immediate results, these will be important in later sections of the report.

Definition 2.4. A conceptor C is called hard if all its singular values satisfy $s_i \in \{0,1\}$.

Corollary 2.1. *If a conceptor C is hard it is invariant under aperture adaptation*

Proof. Follows from Definition (2.4). Suppose a conceptor $C(R,\alpha)$ is hard. Additionally, let $\gamma \in \mathbb{R}$ be an arbitrary scaling factor, we show that $\phi(C,\gamma) = C$. Write the SVD of C as $C = USU'$, the aperture shifted conceptor can be written as $\phi(C,\gamma) = \phi(USU',\gamma) = U\phi(S,\gamma)U'$. The aperture shifted singular values $\phi(S,\gamma)$ can be obtained using Equation (2.10). Since C is hard, by Definition (2.4), $s \in \{0,1\}$ for all singular values on the diagonal of S. Suppose $s = 0$, by Equation (2.10) $\phi(s,\gamma) = 0 = s$. Additionally, if $s = 1$ by the same line of reasoning $\phi(s,\gamma) = 1 = s$. So, $\phi(S,\gamma) = S$, implies that $\phi(USU',\gamma) = USU' = C$. Stated equivalently, C is invariant under aperture adaptation. □

Corollary 2.2. *A conceptor C is hard iff it is a projection matrix*

Proof. if Suppose $C(R,\alpha)$ is hard. To show C is a projection matrix it is sufficient to show $C^2 = CC = C$. Consider the SVD of $C = USU'$, squaring C we obtain $C^2 = CC = USU'USU'$. Since U is an orthogonal matrix, we have that $C^2 = US^2U'$. Recall that C was assumed to be hard, meaning the diagonal of S only contains unit or zero values, from which it is obvious that $S^2 = S$. Therefore, $C^2 = C$, means that C is a projection matrix.

only if Suppose C is a projection matrix. If C is a projection matrix it satisfies a different definition of a conceptor. That is, C is a positive semi-definite matrix with singular values in $[0,1]$, which means C is a conceptor matrix. Moreover, since C is also a projection matrix, its eigenvalues are either unit or zero, hence by definition of a singular value, so are the singular values of C. From this, it follows that C is also hard.

□

2.2.4 Boolean operations (negation)

One of the motivations to study conceptors was their ability to construct symbols from neural data. There are different forms of symbol manipulations defined on conceptors ranging from fuzzy- to Boolean logic. We will not cover in depth the semantics associated with the Boolean logic of conceptors. However, one operation is important to mention for our concerned region of study, that is the negation operation.

Definition 2.5 (Negation Operation)**.** If C is a conceptor matrix, then its negation $\neg C$ is defined as;

$$\neg C = I - C \tag{2.11}$$

It is easily that the negation operation flips the singular values of a conceptor. Since, $\neg C = I - USU' = U(I-S)U'$, where USU' is the SVD of C. We will later see that in some cases this will prove to be useful.

2.2.5 Conceptor-based classification

Another application of conceptors is to use them as a method for classification, more specifically to recognise dynamical patterns. Unlike many other classifiers, conceptor-based classifiers can be incrementally trained and extended. This means that new classes can be learnt by the system without having to retrain the entire classifier. Central to conceptor classification is the fact that when an ESN is driven on an input signal, its dynamics are contained in an area of the state space. If the reservoir is again driven on a signal from the same class, it is expected that the state is contained in the same area.

2.2.6 Classifier formalism

To build the classifier we assume some form of a dynamical system is available, that can be an ESN, LI-ESN or LSM. Also, we must have a labelled dataset that contains n distinct prototypes. The assumption for conceptor based classification is that inputs belonging to the same prototype come from the same stationary stochastic process, training and testing inputs are realisations of these stochastic processes. To start classification we need a conceptor for each class j ($j = 1, \ldots, n$), the procedure to obtain these conceptors is described in (Jaeger, 2014). To classify an input signal a network response vector z is constructed. For each class a positive evidence $E^+(z,j)$ value is computed, the positive evidence value is defined by:

$$E^+(z,j) = z^{'} C^j z \tag{2.12}$$

To classify the input, the class belonging to the highest positive evidence value is selected, meaning that the classifier judgement $\hat{j}$ is obtained by computing:

$$\hat{j} = argmax_j \; E^+(z,j) \tag{2.13}$$

2.3 Leaky-Integrator ESNs

Before we continue to frame the theory of conceptors in continuous time it is imperative to introduce a model that deals with continuous time. We discuss the model that is closest to the discrete-time ESN. In essence, transferring to continuous time dynamics is moving from discrete update rules to a system of ODEs. The leaky-integrator ESN dynamics (Jaeger et al., 2007) is defined by an equation similar to:

$$\dot{x} = \frac{1}{\tau}(-Ax + f(Wx + W^{in}u + b)) \tag{2.14}$$

where $x(t) \in \mathbb{R}^N$ a vector containing the activation of all neurons at time t, $b \in \mathbb{R}^N$ a bias vector and $\dot{x} = \frac{dx(t)}{dt}$ the time derivative of the network activation. The matrix $W \in \mathbb{R}^{N \times N}$ contains the weights defining the strength of connections between neurons. The vector $u(t) \in R^K$ is the input signal to the network and $W^{in} \in \mathbb{R}^{N \times K}$ the input connection matrix. The scalar, $\tau > 0$ a time constant, adjusting the value of τ changes the speed of dynamics of the ESN. Finally, $A \in \mathbb{R}^{N \times N}$ a diagonal matrix which contains the leaking rates for individual neurons on its diagonal.

2.3.1 Approximating LI-ESN dynamics

In the previous paragraph, the LI-ESN was introduced as a system of differential equations. Since the system was introduced as an ODE it must be numerically solved to obtain an instance of the network. There are many different ODE solvers available, notably Euler's method. One advantage of Euler's method is that it allows us to express an approximation of Equation (2.14) in a discrete fashion (for a comprehensive review of Euler's method see Biswas et al. (2013)). If we apply the Euler discretisation on Equation (2.14) and set the discretisation value δ to 1 we obtain the following expression:

$$x(n+1) = \frac{1}{\tau}((I-A)x(n) + f(Wx(n) + W^{in}u(n+1) + b)) \quad (2.15)$$

Where commonly the symbol R is introduced as the retainment rate of the network defined by $R = I - A$. Equation (2.15) is important because it allows us to approximate the network dynamics of Equation (2.14) by an expression similar to the original ESN dynamics as expressed by Equation (2.1). It is noted that we set the the discretisation value δ to 1 for simplicity. In practical situations this would yield extremely poor approximations of the system dynamics.

2.3.2 Echo state property

The ESP was already discussed in a previous section. It is known that conditions for the ESP in leaky-integrator ESNs are different from regular ESNs (Jaeger, 2002). In contrast to the discrete ESN, the ESP for the leaky-integrator ESN is also dependent on τ, A and δ, besides the spectral radius of the internal matrix W. Since Equation (2.14) will eventually be solved on a digital machine, it must be discretised, the step-size between evaluations is represented by δ (note that δ is the step-size applying Euler-discretization). The Euler discretisation is mentioned because it allows us to express conditions for the ESP algebraically. Generally, the network as in Equation (2.14) has the ESP if the matrix $\tilde{W} = \frac{\delta}{\tau}W + (1 - a\frac{\delta}{c})I$ has a spectral radius $|\lambda_{max}| < 1$ (for a proof see Jaeger et al. (2007)), but the ESP for LI-ESNs is also connected to properties of the input signal.

2.3.3 Some observations

Transferring to continuous-time dynamics has several consequences. The first one is that we can no longer explicitly act on the state of the network $x(t)$, rather we can manipulate the state indirectly by acting on its derivative $\dot{x}$. The following is a summary of the key differences between the discrete and continuous models;

Remark. *For a continuous-time leaky-integrator ESN as in Equation* (2.14)*;*

- *Dynamics are given by a system of ODEs.*
- *New variables: leaking rate A, time constant τ.*
- *We can only act on $\dot{x}$.*

2.4 Liquid State Machines

We now discuss a recurrent spiking neural network architecture, the liquid state machine. The LSM (Maass et al., 2002) is introduced to explore what role conceptors can play in more 'biologically plausible' systems. Precisely, we will use the conceptor classification mechanism in combination with an LSM and examine how the pattern regeneration problem can be translated into a spiking context. To discuss the liquid state machine we will first become familiar with the leaky integrate-and-fire (LIF) neuron model, which introduces spikes as characteristic events to communicate between neurons. After we have treated the LIF mechanism we introduce the LSM.

2.4.1 The LIF neuron

Spiking neurons differ from other models discussed in this report. The dynamics of spiking neurons incorporate an action potential, similar to biological neurons, this change has two major consequences. Foremost, it means that neurons only 'communicate' when a spike event occurs. This is unlike non-spiking neurons where connected neurons influence each other's activation at each time-step/point. Furthermore, it (for most neuron models) means that an individual neuron becomes a non-linear dynamical system that when connected forms an even more complex system.

The integrate-and-fire is a computationally cheap model class of a biological neuron (Abbott & Dayan, 2005). In the LIF model, a neuron is modelled as an electrical circuit. The membrane of the neuron is modelled as a resistor R and capacitor C that are connected in parallel, the capacitance and resistor are fed an input current $I(t)$. These electrical components model the hypothetical electrical properties of the neuron. To generate spikes the model assumes that when the membrane potential of a neuron V reaches some threshold μ (i.e $V \geq \mu$) the neuron emits a spike. It is readily derived that the dynamics of the model are governed by Equation (2.16), a single scalar linear first order differential equation, where $\tau = RC$ and V_{rest} the resting potential of the model.

$$\tau \frac{dV}{dt} = V_{rest} - V(t) + RI(t) \tag{2.16}$$

While Equation (2.16) describes the dynamics of the membrane potential, this differential equation alone is not enough to generate spiking behaviour. As stated earlier, the LIF neuron emits a spike when the membrane potential V reaches the spiking threshold μ. The refractory period is not implemented by the equations of the model, but rather as a computational rule. The membrane potential of a cell remains constant during a period of t_{rf} after which the potential is set to V_{rest}, the resting potential of the cell. So, the full LIF model is the combination of Equation (2.16) that describes how the membrane voltage evolves, membrane threshold μ that describes when a spike occurs and the implementation of a refractory period.

To connect LIF neurons some form of synaptic dynamics must be defined, a model that describes how a spike is conveyed from the pre- to post-synaptic neuron. In this report, LIF neurons are connected by a weight $w_{ij} \in \mathbb{R}$. If the 'pre-synaptic' neuron i spikes, it will change the membrane potential of the 'post-synaptic neuron' j by adding $w_{ij}\ mV$. Thus, when we are dealing with a connected population of neurons we are also dealing with a weight matrix W that describes the changes of the post-synaptic voltages when a spike occurs.

2.4.2 Analysis of the neural signal

Since the LIF model treats spikes as uniform events, the behaviour of a single neuron during a period of length T is defined by the set $\{t_i\}_{i=0..a}$, where t_i represents the time of the i-th spike during period T. The behaviour of a LIF neuron can also be represented as a sum of Dirac delta functions.

$$\rho(t) = \sum_{i=1}^{n} \delta(t - t_i) \tag{2.17}$$

To characterise the state of a neuron we will occasionally use properties of the spike train it generates, specifically, its firing rate as defined by:

$$r = \frac{n}{T} = \frac{1}{T} \int_0^T \rho(\tau) d\tau \tag{2.18}$$

Where r represents the firing rate of the neuron during a period of length T and n is the number of spikes during period T.

2.4.3 LSM basics

The liquid state machine (Maass et al., 2002) (LSM) is a computational framework to understand computation in neural (micro-)circuits. The LSM is often seen as a spiking equivalent to the echo state network since they both make use of the RC framework. In contrast to other computational frameworks such as the finite state machine and the Turing machine, computation in liquid state machines is real-time and continuous, making them appealing to study as models for neural computation. Maass et al. (2002) showed that the LSM has universal computational power for analogue, 'fading memory', real-time functions if it satisfies the approximation property (AP) and separation property (SP). The SP is concerned with the trajectories of neurons in the reservoir, it states that different input signals must lead to different trajectories for the spiking neurons in the reservoir. The AP is concerned with the properties of the readout map, it states that the readout map can distinguish the relevant set of liquid states and adapt these states into the required output signal.

Similar to the ESN, the LSM is a mechanism to compute output signals $y(t)$ from input signals $u(t)$, where here both u and y are continuous-time signals. Similar to the ESN, an LSM M generates for every point in time t a state $x^M(t)$ that should accurately represent the response of the LSM to the input history of the presented signal. The state of an LSM is a continuous signal, similar to the state of an LI-ESN. Moreover, similar to the ESN the liquid state machine computes an output signal by constructing a memory-less readout map that uses the state of the system to compute the output signal.

Formally, an LSM M consists of two operators, a liquid filter L^M and a readout map f^M. The liquid filter is implemented by a recurrently connected population of spiking neurons such as LIF neurons together with an operation that defines a state on the SNN. The neurons in the liquid filter are called liquid neurons. The liquid filter computes for each point in time a liquid state based on the SNN, formally:

$$x^M(t) = (L^M u)(t) \tag{2.19}$$

In many cases, the liquid filter is implemented by some form of time-averaging performed on the spike trains that are produced by the liquid neurons when the SNN is presented with an input signal. This could either be done by an exponential decay filter or binned time averaging.

The second operator of the LSM is the readout map f^M that computes an output signal based on $x^M(t)$, hence the output of the LSM can be represented as:

$$y(t) = f^M(x^M(t)) \tag{2.20}$$

The definition of the readout map allows for a broad range of objects that could be used. For instance, it can be implemented by linear regression, Bayes classifiers or a whole new spiking neuron population. Similar to an ESN only the readout map is learnt for a single learning task while the reservoir remains unchanged.

2.5 Continuous-time conceptors

The first section of this report introduced conceptors constructed from echo state networks and discussed their most important properties in the context of pattern regeneration. We concluded that conceptors characterise how the state of the ESN evolved when it is fed a signal by dampening certain regions of the state space. This was achieved by directly acting on the internal state of the network using a conceptor matrix C. The second section introduced the leaky-integrator echo state network as a model that bridged the gap between discrete and continuous dynamics. It was noted that dynamics were described by systems of ODEs instead of update rules. Consequently, the ability to explicitly act on the internal state of the network was lost. In this section results from the previous two sections are combined to obtain expressions and ideas about the task of pattern regeneration in continuous time. First, the problem is formally defined after which two possible solutions are studied. To do this, the notions of refractive and stable trajectories are introduced.

2.5.1 Pattern regeneration for hard conceptors

This section introduces pattern regeneration as a formal setting for hard conceptors. It introduces the notions of a stable and refractive trajectory to formalise pattern regeneration for hard conceptors in an attempt to get a better understanding of pattern regeneration in general. In the words of David Hilbert "He who seeks without a definitive problem in mind seeks for the most part in vain" (MacHale, 1993). So far, we have not given a formal statement of the problem at hand. This section is concerned with formally stating what properties we want the controlled systems to have. The term control generally refers to creating some desired behaviour in a system. In our case, the desired behaviour is regenerating a signal from a loaded reservoir. In the discrete case, this was achieved by restricting the internal state to an area $\tilde{U}$ of the state space. By restricting the state, the response signals show oscillations similar to those obtained when feeding the network the input signal. These oscillations generated an output signal which is similar to the desired output signal. To formalise the problem at hand we introduce two definitions which together lead to a problem statement. We first introduce the notion of a refractive trajectory.

Definition 2.6 (Refractive trajectory)**.** Given a continuous time ESN as in Equation (2.14), a signal $u(t)$ with characteristic subspace $\tilde{U}$, conceptor $C(R, \alpha)$ and the trajectory $x(t, x_0)$ with initial condition $x_0 \in \mathbb{R}^N$ and time t. A trajectory $x(t, x_0)$ is refractive if for some time $T \geq t$, $x(T, x_0) \in \tilde{U}$.

Intuitively, a trajectory is refractive if it 'reaches' an internal state that is characteristic of the corresponding input signal at some time. The term refractive is borrowed from optics (as is common in conceptor literature). It refers to the property of light (or waves in general) being bent when it is propagated from one medium to another. The controller should bend the internal state of the network towards a state that is characteristic of the input pattern that is to be retrieved. Besides actually reaching a characteristic state, we also want the state to remain characteristic over time. This is formalised by the notion of a stable trajectory.

Definition 2.7 (Stable trajectory)**.** Given $x(t, x_0)$ and some $T \in \mathbb{R}$, s.t $x(T, x_0) \in \tilde{U}$ and signal $u(t)$ with characteristic space $\tilde{U}$. $x(t, x_0)$ is called stable if for any time $t' > T$ we have that $x'(t', x_0) \in \tilde{U}$.

For hard conceptors, pattern regeneration is the problem of obtaining trajectories that are both stable and refractive, as formally stated:

Problem 2.1 (Pattern Regeneration)**.** Given a continuous time ESN loaded with signal $u(t)$ with characteristic area $\tilde{U}$, conceptor $C(R, \alpha)$ and state $x(t, x_0)$ at time t. Change the dynamics of $x(t, x_0)$ such that the resulting trajectory is both stable and refractive.

To state the problem informally 1) when using the conceptor as a control mechanism the state of the network should converge to a state in $\tilde{U}$ after some time, meaning it should reach a characteristic internal state for the input signal associated with the conceptor. Additionally, 2), once such a state is obtained we want that the state remains characteristic over time. Now that we have framed the problem in a rigorous setting we can begin to study possible solutions. However, one might raise the question, why even bother constructing these definitions, why not use the same controller in continuous time? The reason why becomes apparent once we study how the discrete controller acts in continuous time. As we will see, we have no guarantee for controlled trajectories to be either refractive or stable.

2.5.2 Discrete control does not yield refractive trajectories

Recall that the autonomous pattern regeneration rule for discrete ESNs is given by:

$$x(n+1) = C^j f(\tilde{W}x(n)) \tag{2.21}$$

We make two important observations about Equation (2.21). First of all, control is obtained instantaneously, meaning that $\tilde{U}$ is reached in a single update. Additionally, we can act on the state x explicitly. We consider what happens when we apply the discrete controller in continuous time. Interpreting the idea of discrete control in continuous time leads to the following equation:

$$\dot{x} = \frac{1}{\tau} C(-Ax + f(Wx + b)) \tag{2.22}$$

It follows from a simple argument that the resulting trajectory is not refractive. Suppose the trajectory has initial condition $x_0 \in \tilde{U}$, obviously this yields refractive trajectories. Now, suppose $x_0 \notin \tilde{U}$, then we have no guarantee of ever reaching $\tilde{U}$. Since, any output of Equation (2.22) is a point in $\tilde{U}$. Therefore, the obtained differential will never 'point' towards $\tilde{U}$. Hence, the resulting trajectory is not refractive.

Now that have seen that trajectories resulting from Equation (2.22) cannot give rise to pattern regeneration, since they are not guaranteed to result in refractive trajectories. This is the case because Equation (2.22) does not dampen the regions of the state-space that are not characteristic of the state. Therefore, we need to construct methods that not only evolve the state (similar to the discrete case) characteristically but also project the state into the characteristic area. We now consider two methods that achieve this.

2.5.3 Projection control

After we derive a possible solution for hard conceptors we generalise it for arbitrary conceptors. Recall that hard conceptors are projection matrices, meaning that $\tilde{U}$ is a proper linear subspace. This means that when C is hard we can construct orthogonal projections from an arbitrary state to an arbitrary point in $\tilde{U}$. Specifically, the point in $\tilde{U}$ which is closest to $x(t)$. This closest point can be found using orthogonal projections in the direction of $\tilde{U}$. An orthogonal projection of a vector is defined as $x = \hat{x} + Px$, where $\hat{x} \in \tilde{U}$ and $Px \in \tilde{U}^{\perp}$, where the $\perp$ superscript denotes the orthogonal complement of a vector space. This means that $\hat{x}$ is orthogonal to Px. More importantly, Px is a vector that points from $\tilde{U}$ to $x(t)$. For now, we assume it is possible to construct orthogonal projections of the form $x = \hat{x} + Px$. In this case Px can be used to compute how far the state is from the closest point in $\tilde{U}$. In essence, we use the orthogonal projection to construct an error signal for each neuron. This error signal expresses the distance between neural response and the characteristic neural response. The error signal is an indication as to what activation of neurons should be dampened by the controller. It is important to note that P is only of use when $\tilde{U} \neq span(\mathbb{R}^N)$, as stated by the following corollary:

Corollary 2.3. *$x \in \tilde{U} \to Px = 0$ and $x \notin \tilde{U} \to Px \neq 0$.*

Since if this is not the case, $\tilde{U}$ spans the entire state space, hence any state is a point in $\tilde{U}$. Consequently, Corollary (2.3) states that Px will always be zero, thus the controller will not alter the dynamics of the network. The orthogonal projection can be used to construct a control rule that is refractive under the assumption of a hard conceptor. One of many ways to construct such a controller is to perform linear proportional control on Px in addition to the discrete control rule. This means that the projection matrix P is coupled with a constant $K \in \mathbb{R}^{\geq 0}$ and added as an additive term in the differential equation. Hence, the state update equation becomes:

$$\dot{x}(t) = C(-Ax(t) + f(Wx(t))) - KPx(t) \quad (2.23)$$

We point out two shortcomings of this section. First, at this point, no formal proof is given that Equation (2.23) solves the pattern regeneration problem. Rather, Section 4.1 will be presented to highlight the derived theory. Additionally, we do not provide procedure to find a matrix P such that $x = \hat{x} + Px$.

2.5.4 A geometric interpretation of pattern regeneration for hard conceptors

Another reason why we consider the case where C is hard is because it has a geometric interpretation, which is displayed in Figure 2.1. This geometric perspective illustrates both the essence of Problem (2.1) and the reason why the discrete rule does not solve it (in continuous time). For hard conceptors interpreting the control task geometrically can be understood as letting the state x converge to a point in the red plane (refractive), which is the characteristic subspace of the conceptor $\tilde{U}$. And, once having obtained such a state not leaving the red plane anymore (stable). In discrete time, the red plane can be reached in a single step, meaning that it is possible to jump from x to a point in the red plane instantaneously. However, in continuous time we have to 'continuously' project the trajectory of x to a point in the red plane. If the state (blue dot) is not already a point in this plane we can define a vector which is the orthogonal projection of x onto $\tilde{U}$, this coincides with the vector Px.

2.5.5 Negation control

The previous section discussed the geometry of the control problem for hard conceptors, hence the

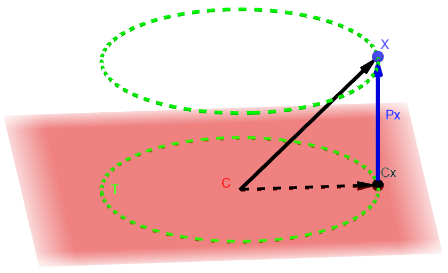


Figure 2.1: Geometric perspective pattern regeneration

arguments constructed under the assumption of hard conceptors do not necessarily hold for arbitrary conceptors. In this section, a similar method is derived for arbitrary conceptors, which is then used to construct a more general controller. Recall that for the previous method an "error" signal was constructed by computing the distance from the current state to the closest point in $\tilde{U}$ (for each neuron), this was done using orthogonal projection. Since $\tilde{U}$ was a linear subspace, this resolves to finding the solution to a least-squares solution. In essence, both methods rely on finding a point which is in the characteristic subspace $\tilde{U}$ and constructing an "error" signal based on the difference between the current state and this point. For the previous method, this point was the point in $\tilde{U}$ closest to x. However, we can find another point in the characteristic area using the conceptor, namely Cx. We then again can obtain an error signal $(I - C)x = \neg Cx$, meaning that the negation conceptor of C also provides an "error" signal, the next section states a condition under which the two error signals are equal. Similar to the previous method, when we have a state which is in the 'characteristic' area corresponding to the input signal, the control signal is nearly zero. The controller that works for an arbitrary conceptor is then obtained in a similar method, meaning that we add a control constant which is added as an additive term in the differential equation in combination with the discrete control rule, doing so we obtain:

$$\dot{x}(t) = C(-Ax(t) + f(Wx(t))) - K\neg Cx(t) \quad (2.24)$$

2.5.6 Obtaining discrete from continuous control

We now shift our focus to a different but also important question. We investigate how the discrete and continuous regeneration dynamics are related. Specifically, we will try to obtain Equation (2.24), starting from the discrete-time approximation of the LI-ESN. To determine whether this is possible, we begin by obtaining a discrete version of the network dynamics starting from Equation (2.14). After which we apply the known discrete regeneration rule. We then attempt to rewrite this expression back to a system of differential equations. We start by recalling the definition of the time derivative.

$$\dot{x}(t) = \frac{dx(t)}{dt} = \lim_{\epsilon \to 0} \frac{x(t+\epsilon) - x(t)}{\epsilon} \tag{2.25}$$

We substitute the definition of the derivative into Equation (2.14) (assume $\tau = 1$ for simplicity), and solve for $x(t+\epsilon)$.

$$x(t+\epsilon) = \epsilon(-Ax(t) + f(Wx(t) + W^{in}u(t))) + x(t) \tag{2.26}$$

Dynamics described by Equation (2.26) can be interpreted as making the dynamics discrete. The dynamics are now expressed as a discrete-time approximation of the ODE, that is if ϵ is not infinitesimal. Let $u(t) = 0$, hence $W^{in}u(t) = 0$ and we plug in the conceptor similar to Equation (2.21).

$$x(t+\epsilon) = C(\epsilon(-Ax(t) + f(Wx(t))) + x(t)) \tag{2.27}$$

Finally, by plugging (2.27) into (2.25) we obtain:

$$\dot{x}(t) = C(-Ax(t) + f(Wx(t))) - \lim_{\epsilon \to 0} \frac{\neg C}{\epsilon} x(t) \tag{2.28}$$

The interesting term in this final equation is $\frac{\neg C}{\epsilon} x(t)$. This limit tends toward ∞ as ϵ tends to 0. This property has a nice interpretation. We know that the discrete control was instantaneous, meaning that it was possible to obtain a characteristic state in a single update. If we interpret this in a continuous setting, instantaneous control is only possible if we are dealing with an infinitely large differential towards $\tilde{U}$. In other words, instantaneous pattern regeneration in continuous time is only possible when the initial condition is a characteristic state, else it requires an infinitely large differential towards $\tilde{U}$.

The final paragraphs of this section are dedicated to giving some more intuition and interpreting the role of parameters in the derived methods. We have seen that for both methods a control parameter K was introduced besides the preexisting parameter of aperture α. We dedicate a section to the interpretation of the new constant K. At the time of writing, we cannot give a complete description of effects of aperture variation on Equation (2.24), some remarks are given in the discussion section.

2.5.7 Interpreting K

The control parameter K was introduced in the previous section and has a relatively straightforward interpretation. Geometrically, for hard conceptors, K can be seen as a scaling of the vector that points from $\tilde{U}$ towards x, (refer to Figure 2.1 for visual representation). Hence, increasing K grows the effect of the introduced term $K\neg Cx$ during pattern regeneration described by Equation (2.24). To summarise the interpretation of K, the following statements characterise the most important aspects of the control parameter K.

Remark. *Given a conceptor C and $K > 0$.*

- *Adjustments of K are linear.*
- *$K\neg Cx$ decreases with $||Cx - x||$.*
- *K scales the singular values C.*
- *K is a gain factor in proportional control. Making K too large will lead to instable behaviour.*

3 Experiments

We began the previous section to study the pattern regeneration problem in continuous time. We have seen a method that can theoretically achieve this for arbitrary conceptors. In deriving this method we discovered that continuous pattern regeneration asks different things of the conceptor compared to pattern regeneration in discrete time. Besides that, we have seen that this method introduces a new parameter K.

To illustrate and test the theory derived in the previous section we will conduct two computer simulations which serve different aspects of the conceptor theory. Both experiments are also directly derived from the simulations described in (Jaeger, 2014). The first experiment illustrates the validity of Equation (2.24). The second experiment illustrates an LSM classification scheme that uses conceptors.

3.1 Pattern regeneration

Pattern regeneration is the task of retrieving patterns that are stored in a reservoir using conceptors. Pattern regeneration has been the focus of a large part of this report. Performing a simulation allows us to test the derived theory, making sure that the derived methods actually work as expected. It also provides an illustration of the fact that regeneration in continuous time is not instantaneous.

3.1.1 Initialising the reservoir

The simulation implements an LI-ESN with dynamics governed by Equation (2.14), where $N = 200$ and $f = \tanh$, both the input and output size were one ($K = L = 1$). Additionally, the system had a leaking rate of $a = 1$ for all neurons and a time constant $\tau = 1$ was used. The input weights, bias vector and reservoir weights were sampled from $\mathcal{N}(0, 1)$. The internal weight matrix was re-scaled such that it had a spectral radius of $|\lambda_{max}| = 1.5$, the input weights were re-scaled by a factor of 1.2 and the bias vector was re-scaled by a factor of 0.2. The two input signals are defined as $p^0(t) = \frac{1}{2}\sin(\frac{t}{7})$ and $p^1(t) = \sin(\frac{t}{10})$.

3.1.2 Preparing the reservoir

Preparing the reservoir consisted of several different tasks, the readouts have to be computed, the signals must be stored in the reservoir and for each signal, a conceptor must be computed. For both input signals, a simulation was run for $0 \leq t \leq 1500$, solving (2.14) numerically, using the lsoda solver for systems of differential equations. The initial condition was different for each run and sampled from $\mathcal{N}(0, 1)$. For each integer time-point of the interval ($t = 1, 2, \ldots, 1500$) the state of the network was collected, of which the first 500 were discarded. Hence, a single run, yields 1000 sampled states for each pattern. For each pattern, these states were collected into two separate state matrices X^1 and X^2 which were concatenated to obtain a single state matrix $X = [X^1X^2]$. Likewise, two delayed state matrices $\hat{X^1}$ and $\hat{X^2}$ were built, by applying a unit delay to X^1 and X^2, these were concatenated to obtain $\hat{X} = [\hat{X^1}\hat{X^2}]$. Additionally, two matrices P^1 and P^2 were created such that they matched the input for the corresponding state matrix, meaning that $P^i[j] = p^i(j + 500)$. The two matrices were concatenated to obtain one big input matrix $P = [P^1P^2]$. Readouts were computed using ridge regression. Where the regularizer was set to $\lambda = 0.01$.

$$W^{out} = ((XX^{'} + \lambda I)^{-1}XP^{'})^{'} \tag{3.1}$$

Similarly, to load the two signals into the reservoir again ridge regression was used, in this case, the regularizer is set to $\lambda_W = 0.01$.

$$W = ((\tilde{X}\tilde{X}^{'} + \lambda_W I)^{-1}\tilde{X}P^{'})^{'} \tag{3.2}$$

Finally. for both X^1 and X^2 the correlation matrices $R^1 = X^1X^{1'}/L$ and $R^2 = X^2X^{2'}/L$ were computed. Two conceptors C^1 and C^2 obtained by computing Equation (2.8) for both state matrices.

3.1.3 Performing Pattern Regeneration

To perform pattern regeneration Equation (2.24) was numerically solved, using the computed loaded weights and conceptors. The system was solved for the same time interval that was used during training. Additionally, the same time points were collected to compute the retrieved output signal. During regeneration, no reservoir states were discarded as a washout.

3.1.4 Miscellaneous details

As mentioned earlier, to solve the system of ODEs a numerical integrator from the python package SciPy was used, this is the lsoda numerical integrator. The numerical algorithm the lsoda solver uses is more complex than the Euler-discretisation commonly used to approximate LI-ESNs.

3.2 LSM conceptor classification

This experiment serves as an idea to implement conceptors in a continuous-time context using spiking neurons. Specifically, we describe a classification procedure and test it on a dataset of 4 classes, each a different type of sine wave. The four classes was based on the following set of signals; $\{\sin(2\pi t), sawtooth(8t), \cos(12\pi t), square(4t)\}$, where *square* indicates that a square wave was used and *sawtooth* that a saw tooth wave was used. The experiment aims to investigate how robust the classifier is to noise. The classifier will 'learn' 4 periodic signals, the classification procedure will consist of tests that manipulate the phase and amplitude of these signals slightly.

3.2.1 BSA algorithm

Since LSMs deal with spiking neurons, we need a method to encode analogue signals into spike trains. To encode signals into spike trains the BSA algorithm was used (Schrauwen & Van Campenhout, 2003). The BSA algorithm uses a finite impulse response (FIR) filter to construct a spike train which represents the analogue signal in the spiking domain. BSA iterates through the sampled signal and computes two different error metrics, it then compares the difference between the errors against a threshold. If the difference meets the threshold, a spike is generated at that time step and the filter is subtracted from the input. The pseudo-code of BSA is added in appendix A.1.

3.2.2 Initialising neurons

The BRIAN2 framework was used to digitally simulate LIF neurons. A liquid filter containing 10 neurons was constructed, each neuron had a refractory period of 3 ms a spiking threshold of 15 mV a time constant of 30 ms and a reset voltage of 13.5 mV.

3.2.3 Initialising synapses

50% of the population was randomly selected to be inhibitory, the rest was excitatory. The neuron parameters of inhibitory neurons were equivalent to excitatory neurons. The probability of a synaptic connection was 0.7. The value of synaptic weights was determined by picking a random number in $[0, 1]$ and multiplying it by a factor of 0.7 if the pre-synaptic neuron was excitatory and 0.4 if it was inhibitory. The unit of all synaptic connections was mV.

3.2.4 Initialising input signals

Each sample was transformed into a spike-train by computing its BSA transform using a pre-existing python library. From the BSA transform exact spike times were extracted. The extracted spike times were then filtered based on a refractory period, the refractory period was set constant at $15ms$. This yielded the final spike trains that served as an input to the liquid filter. The input was injected into all liquid neurons, with a scaling of 1.2 for excitatory and 0.8 inhibitory neurons.

3.2.5 Initialising the classifier

The four classes learnt by the classifier were the signals, $4\cos(\pi t)$, $\sin(5t)$, a saw tooth wave with frequency 8 and a square wave with frequency 1. To initialise the classifier a conceptor was computed for each of these four signals. To do this, for all four signals the filtered BSA transform of this sample was used as the input to the LSM. To compute a positive evidence conceptor, the following procedure is followed. In four independent runs, a filtered BSA transform was fed into the LSM which was run for 5 s. During this period, the spiking activity of the liquid neurons was monitored and collected, obtaining 10 sequences of spike trains $\{s_0, s_1, \ldots, s_{10}\}$. These spike trains were placed into bins of 200 ms, meaning a single run consisted of 25 bins. For each bin the firing of a neuron during that bin was computed, this yields for each neuron a vector such that:

$$x_i = [r(1)\ r(2)\ \ldots\ r(n)] \tag{3.3}$$

Where r indicates the firing rate of the neuron during bin n of the experiment. The individual firing rates were collected into a state matrix. From the four obtained state matrices a conceptor was computed, after which the classifier was considered initialised.

3.2.6 The classification scheme

Having initialised the classifier, a single trial consisted of a signal $p(n)$ obtained by manipulating one of the signals learned by the classifier. First, a random integer was used to decide from what learnt signal the pattern was to be constructed. After a signal was selected it was permutated in two ways, first the amplitude was scaled by uniformly sampling a random number in the interval $[0.8, 1.2]$ and its phase was changed by uniformly sampling a random number in the interval $[0, \frac{\pi}{2}]$. The permutated signal was then sampled using these numbers and its filtered BSA transform was used as an input to the LSM. Again, the activity of the liquid neurons was monitored during a period of $5s$ and a state matrix was computed in the same manner that was described in the previous section. To classify the permutated input signal the following this state-matrix was collected into the vector z such that:

$$z = [x(1)\ r(BSA(p(1)))\ \dots\ x(n)\ r(BSA(p(n)))] \tag{3.4}$$

Where $r(BSA(u(n)))$ denotes the firing rate of the filtered BSA transform of the signal $p(t)$ during bin n of the experiment. Finally, the judgement of the classifier was obtained by using Equation (2.13) on z.

4 Results

4.1 Pattern Regeneration

We first inspect the performance and overall quality of the constructed model. After we make assertions about the model quality we comment on the influence of K during simulations.

4.1.1 Model performance

To evaluate the model performance we perform an analysis by computing an error metric between the driving/goal signal $u(t)$ and the signal $W^{out}x(t)$, where $x(t)$ is obtained using two different versions of Equation (2.24), one for $K = 1.5$ and one for $K = 0$. Note that $K = 0$ means that no control on the trajectory is performed. The error metric that was used is the normalised root mean squared error (NRMSE). To find the best accuracy for a single run both signals were shifted over 16 points and the best NRMSE outcome was used. The error metrics were obtained by running the procedure described in Section 3.1 for independent 5 trials, meaning that for each run a new reservoir was initialised. For each run, the optimal NRMSE was computed, and the optimal NRMSEs were collected and averaged, this yielded an average NRMSE of 0.2 for $K = 1.5$ and an average of 5.4 for $K = 0$. The output signals that were generated during the experiment that were used to compute the NRMSEs are added as an appendix. This is to highlight the significant difference in the quality of pattern regeneration to which NRMSE does not do justice. To state that controlled pattern regeneration using Equation (2.24) (i. e $K \neq 0$) far outperforms the uncontrolled trajectories (i.e $K = 0$) would be an understatement.

Besides performing a formal analysis it is also important to present a more graphical interpretation of the results. Graphical inspection allows us to investigate other parts of the derived theory. Figure 4.1 summarises the most important graphical interpretations and illustrates the effect the newly introduced control term has. Figure 4.1 illustrates four attempts of pattern regeneration (of two different patterns), where for two attempts $K = 0$ and for the other two $K = 1.5$. For the runs where $K \neq 0$, the $K\neg C$ appears to have a stabilising function in the dynamics of the system. It appears that introducing a non-zero control term makes sure the periodic oscillations of neurons are centred around an equilibrium point as if the oscillations are bounded when K has a large enough value. On the other hand, looking at the reservoir states when $K = 0$ no such equilibrium point seems to exist, and the states seem not to have any recurring oscillations around the x-axis. In informal terms, having a large enough value for K seems to quite literally control the neural response signals into trajectories obtained when driving the system on the driving input, as was predicted when deriving Equation (2.24).

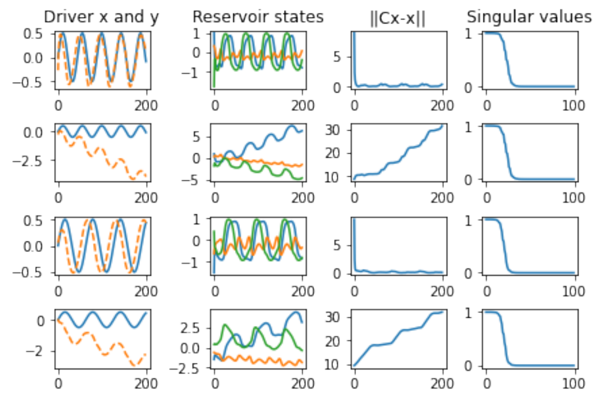


Figure 4.1: Graphical representation of a retrieval attempt. The left column plots the retrieved signal (dashed) and driver signal, and the centre-left column shows three neural response signals. The centre-right column shows a metric used to indicate retrieval success $||Cx - x||$, and the right column plots the singular values of the conceptor used. The plot contains four rows, the first row is controlled regeneration using Equation (2.24) **of pattern** 0**, and the second row is regeneration without the control term $K\neg C$ The third row shows the controlled regeneration goal of pattern** 1**, the last row shows the uncontrolled regeneration of pattern** 1**. The apertures of the conceptors had value $\alpha = 100$, and the value of K was set at** 1.5 **(when applicable).**

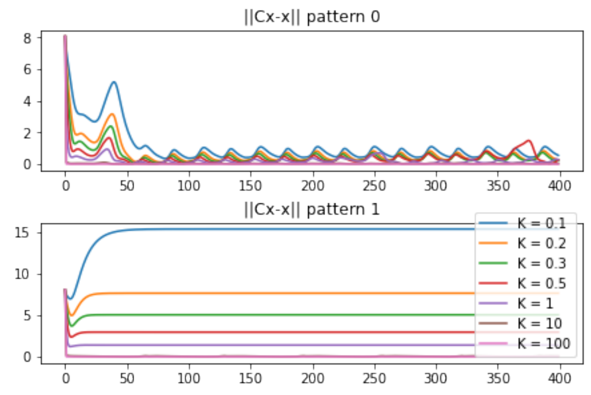


Figure 4.2: Top plot shows the error metric $||Cx - x||$ during pattern regeneration using Equation (2.24) **for several different values of K for pattern** 0**. The bottom plot shows the same for pattern** 1**. The apertures of the conceptors had value $\alpha = 100$, and the value of K was set at** 1.5 **(when applicable).**

Another important observation becomes apparent once the metric $||Cx - x||$ has been computed for the controlled ($K \neq 0$) and uncontrolled runs ($K = 0$). Having a large enough K steers the dynamics of the network towards $\tilde{U}$ while having $K = 0$ only seems to move further away from $\tilde{U}$ over time. The fact that the introduced control term steered the dynamics of the network towards the characteristic area of the pattern was another important theoretical result that is now confirmed by simulations. In the next section, we comment on how different values of K behave when used in Equation (2.24).

4.1.2 Influence of K

Equation (2.24) introduced the parameter K that scales the vector $\neg Cx$. In this section, it is investigated how K behaves when it is varied during retrieval. Results about the behaviour of K are illustrated by Figure 4.2. To construct the figure the reservoir was prepared by the procedure described in Section 3.1. This means that patterns were loaded into the reservoir and the output weights and conceptors were computed. In different runs from the same initial condition different values of K were used in Equation (2.24) to retrieve patterns from the reservoir autonomously, i.e $W^{out}x(t)$ was computed. To indicate how much the term $K\neg C$ constrains the dynamics of the network, the metric $||Cx - x||$ is used.

Generally, Figure 4.2 shows three different behaviours when K is varied.

1. The value of K is too small, and the resulting trajectory is neither refractive nor stable.

2. The value of K is too small, the resulting trajectory is refractive but not stable. Generally, it appears that refractive trajectories that are not stable can result in successful pattern regeneration. It is difficult to reason as to why this is the case. Perhaps characteristic regions are "fuzzy", another reason could be that $||Cx - x||$ is not a perfect error metric. However, the fact remains that when K is too small to yield a stable trajectory it does not mean that pattern regeneration cannot be achieved.

3. The value of K is satisfactory. In this case, the resulting trajectory is refractive and also stable.

All in all, it is hard to predict the quality of the pattern regeneration based on an $(||Cx - x||, t)$ plot such as Figure 4.2. Besides the error metric, the quality of loading turns out to be very important, something that is difficult to access numerically when loading multiple patterns.

We conclude this section by stating some other observations made during the analysis of the simulations. It appears that satisfactory values of K encompass quite a large interval. For instance, if $K = 30$ yields the desired behaviour, so will $K = 20$ and $K = 40$. Additionally, 'speed of convergence' is also not significantly impacted by such changes and is rather determined by the initial condition of the trajectory. It is observed that the speed at which $||Cx-x||$ converges to zero slightly decreases when the error metric becomes smaller, as was expected. Finally, satisfactory values of K appear to be pattern and reservoir dependent, as illustrated by Figure 4.2. This implies that a satisfactory value of K does not necessarily work for a different pattern, nor for a different reservoir loaded with the same pattern.

4.2 LSM conceptor classification

The classification procedure as described in the previous section was independently run 5 times, initialising a new LSM for each trial. The average misclassification rate obtained was 12 on a series of 50 queries, meaning that the classifier had an average classification accuracy of 76%. Note that this result could be optimised by several different procedures, such as aperture optimisation, introducing negative evidence values, BSA optimisation and other hyperparameter optimisation. However, the goal of this experiment was not to obtain a state-of-the-art result using conceptor classification (for a review on classification optimisation see Jaeger (2014)). Rather, it was to investigate whether the classification scheme outlined can also transfer to other types of reservoirs such as an LSM. Moreover, it also illustrates whether or not rate coding would be a suitable modelling approach for constructing conceptors in spiking contexts. Although a more rigorous analysis of the classifier is required, preliminary results suggest that the outlined classification scheme is a suitable candidate for classifications in LSMs.

5 Discussion

In the final part of the report, we reflect on the results obtained during the theoretical and practical sections. After the results are put into a broader perspective some notable directions for future work are discussed.

5.1 Continuous-time pattern regeneration

This report started with the extension of the pattern regeneration problem to continuous-time models such as the LI-ESN. For discrete-time ESNs pattern regeneration was relatively simple, we argue that for continuous-time models it is not as simple. Perhaps the most important difference between continuous-time and discrete-time pattern regeneration is that in the former it is impossible to explicitly act on the state of the network, while this is possible in the latter. This fundamental discrepancy led us to define the notions of stable and refractive trajectories and ultimately to Equation (2.24). While deriving said equation, we re-established the interpretation of hard conceptors. Moreover, deriving control methods for continuous-time models a new parameter K was introduced which has a clear interpretation. Considering everything introduced in this section, we look back on several interesting results, notably Equations (2.24) and (2.28). We now illustrate some aspects where the derived theory can be improved upon.

A noteworthy remark arises from the statement of Problem (2.1). While pattern regeneration is formally stated as a problem it is difficult to analytically verify whether trajectories will be stable and/or refractive and solve the control problem. Also, the statement of Problem (2.1) assumes a perfect loading procedure, which is very rare in practical settings. So, while the pattern regeneration problem is stated formally, it is difficult to judge how useful it will be in analytical situations. In practical situations, the statement of Problem (2.1) was sometimes useful, but certainly was not a perfect prediction of the quality of regeneration.

Finally, we note that the additive terms in Equations (2.24) and (2.23) are added to ensure that the trajectory reaches the characteristic area of the input signal. We would like to stress that these terms are not unique and one could think of many other objects that will yield the same behaviour. There are many other possible controllers that one could construct that would also be able to that steer the dynamics of the trajectory towards the characteristic area, think of PID controllers, neural networks or kernel methods. However, these more complicated controllers do not make use of the conceptor framework as studied in this text. Simply stated, this remark illustrates that conceptors can be used as controllers in high-dimensional non-linear dynamical systems, which is remarkable considering their simplistic nature.

5.2 LSM conceptor classification

The second contribution of this text is aimed at making a simulation using conceptors in LSMs. Specifically, the classification mechanism of conceptors was implemented using a liquid state machine. The result obtained by the classifier was satisfactory but not state of the art. Additionally, the result obtained was not as good as other known conceptor applications, this could have several reasons. First of all, not much effort was put into the optimisation of the classifier meaning it could be that an optimisation yields a significant improvement in accuracy. Besides that, the classification task at hand is not comparable to those found in the literature. Finally, the classifier architecture was much simpler than those in the literature. Therefore, future work should investigate the optimisation of conceptors in combination with spiking reservoirs. This could be done by extending pre-existing techniques such as negative evidence values and aperture optimisation to spiking reservoirs.

5.3 Future work

5.3.1 Leading C in Equation (2.24)

One important observation about Equation (2.24) pertains to its leading C. It can be argued that this leading multiplicative term could be removed simplifying the equation to:

$$\dot{x(t)} = -Ax(t) + \tanh(Wx(t)) - K\neg Cx(t) \quad (5.1)$$

This expression can be simplified to:

$$\dot{x(t)} = \tanh(Wx(t)) - x(t)(A + K\neg C) \quad (5.2)$$

It would be interesting to see whether this simplified expression can be used to do pattern regeneration for two reasons. First of all, the simplified expression is more elegant and arguably more transparent. Additionally, it removes a symbol from Equation (2.24), which is an improvement from a computational perspective.

5.3.2 Rigorous definition of $\tilde{U}$

An important addition to the theory derived in this report would be to construct a mathematically rigorous definition of $\tilde{U}$, such that the definition also makes sense for arbitrary conceptors. Currently, Definition (2.2) only makes sense in the context of hard conceptors that are not the identity. This is because $\tilde{U}$ is supposed to formalise the notion of an area of the state space that is covered when driving the system on an input signal. If this area does not vary in all dimensions of the state space, Definition (2.2) makes sense, since it provides some information about the said area. However, if this area varies in all dimensions of the state space $\tilde{U}$ does not provide any information about the area covered and could be interpreted that the area covers the entire state space, while this is certainly not the case. Therefore, a more suitable definition of $\tilde{U}$ should not rely solely on operators from linear algebra. Rather one objects that have a graded concept of being in the area of $\tilde{U}$.

5.3.3 More elaborate simulations

Another line of future work could be to improve the simulations presented in this report and construct new simulations that can further illustrate the theory of conceptors in continuous time with or without spiking neurons. It would be interesting to see how the performance of discrete and continuous pattern regeneration differs from one another and what factors influence the performance, to determine these facts further study is required.

5.3.4 Interpreting aperture

In contrast to K, the value of aperture in Equation (2.24) is not completely understood at the time of writing. Therefore we, cannot provide a full characterisation at this time, we will however attempt to describe some observations. Again for hard conceptors, the interpretation is simple, since hard conceptor are invariant under aperture adaptation, meaning that Equation (2.24) is independent of α. For not hard conceptors, it is important to state one observation, especially for Equation (2.24). It is the fact that Equation (2.24) contains both the negation and the un-negated form of a conceptor in the same equation. At the time of writing, it is not understood how changing the aperture of the conceptor in this equation influences the dynamics of Equation (2.24). Since aperture plays such an important role for nearly all conceptors, it is of interest to investigate the behaviour of aperture in Equation (2.24).

5.3.5 LSM pattern regeneration

We conclude this report by hypothesising about two methods that could achieve pattern regeneration in a liquid state machine. Recall that pattern regeneration is the problem of retrieving a signal from a loaded reservoir in the absence of a driving input. For ESNs and LI-ESNs pattern regeneration was achieved by running the system autonomously using the conceptor to constrain the network dynamics to some area of the state space. We now consider what this would mean if we are dealing with a spiking reservoir. Especially, what Problem (2.1) entails in a spiking context.

The first consideration about pattern regeneration in LSMs is connected to the abstraction mechanism of the LSM formalism and in particular Equation (2.19). The definition of the liquid state as presented in this paper does not yield a complete description of the internal dynamics of the system but rather provides an approximation of the current state. Additionally, abstracting away from the formal state of the system of the liquid neurons has another consequence. Given Equation (2.19) it is impossible to directly alter the dynamics of the network since $x^M(t)$ is only an abstraction from the actual state of the dynamical system. Rather, one must alter the dynamics of the LSM directly by acting on the underlying dynamical system instead. So, to do pattern regeneration (or any other control task) in an LSM, we can only use the liquid state indirectly and must act on components of the dynamical system to steer the reservoir dynamics. We now discuss two methods that act on different components of the liquid filter, one operates directly on the membrane voltage while the other acts on the spiking threshold of liquid neurons. In essence, both methods perform some form of dampening of certain regions of the state space as is expected when discussing pattern regeneration.

The first mechanism manipulates the spiking threshold of the liquid neurons to dampen certain neural responses, as a consequence making the spiking threshold of a liquid neuron a dynamic instead of a static component of the SNN. We already saw how rate-coding can be used to construct bins to obtain a 'discretised' LSM, we also were able to compute conceptors based on the rate-averaged bins. We suggest that changing the threshold voltages based on the activity of a neuron during a discrete bin can give rise to pattern regeneration. To run this 'threshold-based' pattern regeneration, suppose at $t = 0$ an LSM is loaded with signal $u(t)$ and conceptor C has been computed. Additionally, let $v \in \mathbb{R}^N$ be the vector containing the thresholds of all liquid neurons. After every time interval δ, a new vector of thresholds are obtained by computing $v_{new} = Cx^M(t)v$, or a similar expression. Dynamically changing the spiking threshold based on the current state of the network adds a state variable to every individual neuron, namely the threshold itself can now be described as a function of the liquid state $x^M(t)$.

The second mechanism is more in line with methods we have already presented such as Equations (2.24) and (2.23). These methods have the property that the conceptor acts directly on x or $\dot{x}$ providing a manner that influences the dynamics of the system. One could suggest that a conceptor could act directly on the state of the LSM, meaning that the conceptor influences the membrane voltages of liquid neurons. Arguably, directly acting on the membrane voltage of liquid neurons is more transparent and similar to the equations derived in this report, it is however not straightforward how to compute a conceptor in this context. For this method one could think of several approaches to the computation of the state-matrix X, using rate-based coding, spike-time dependant coding and directly having the membrane voltage function as a state vector. Future research could focus on identifying mechanisms that achieve pattern regeneration in LSMs, truly putting the study of conceptors in a more general computational setting.

References

Abbott, L. F., & Dayan, P. (2005). *Theoretical neuroscience: computational and mathematical modeling of neural systems.* MIT press.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press. doi: 10.1093/acprof:oso/9780195324259.001.0001

Biswas, B., Chatterjee, S., Mukherjee, S., & Pal, S. (2013). A discussion on Euler method: A review. *Electronic Journal of Mathematical Analysis and Applications*, *1*(2), 2090–2792.

He, X., & Jaeger, H. (2018). Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on learning representations.*

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, *148*.

Jaeger, H. (2002, 01). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. *GMD-Forschungszentrum Informationstechnik, 2002.*, *5*.

Jaeger, H. (2014). Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*.

Jaeger, H. (2021, jul). Towards a generalized theory comprising digital, neuromorphic and unconventional computing. *Neuromorphic Computing and Engineering*, *1*(1), 012002. Retrieved from `https://doi.org/10.1088/2634-4386/abf151` doi: 10.1088/2634-4386/abf151

Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, *20*(3), 335–352.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human–Computer Interaction*, *12*(4), 391–438.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

MacHale, D. (1993). *Comic sections: The book of mathematical jokes, humour, wit, and wisdom.* Boole Press.

Qian, G., & Zhang, L. (2018). A simple feedforward convolutional conceptor neural network for classification. *Applied Soft Computing*, *70*, 1034–1041.

Redmon, J., & Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, *abs/1612.08242*. Retrieved from `http://arxiv.org/abs/1612.08242`

Schrauwen, B., & Van Campenhout, J. (2003). Bsa, a fast and accurate spike train encoding scheme. In *Proceedings of the international joint conference on neural networks, 2003.* (Vol. 4, pp. 2825–2830).

Yildiz, I. B., Jaeger, H., & Kiebel, S. J. (2012). Revisiting the echo state property. *Neural networks*, *35*, 1–9.

A BSA algorithm

Algorithm A.1 BSA transform of a sampled analog signal

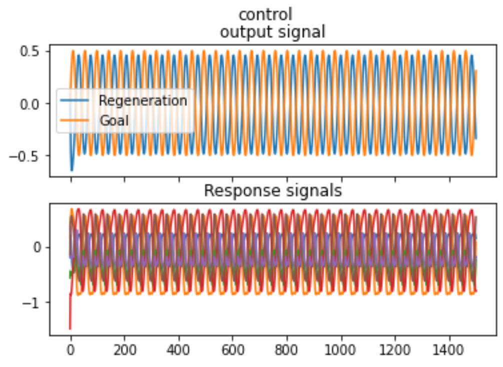
```
Require: $input$, $h$, $threshold \geq 0$
  for $i = 1, \ldots, len(input)$ do
    $error1 \Leftarrow \sum_{k=0}^{M} abs(input[k+i] - h[k])$
    $error2 \Leftarrow \sum_{k=0}^{M} abs(input[k+i])$
    if $error1 \leq (error2 - threshold)$ then
      $output[i] \Leftarrow 1$
      for $j = 1, \ldots, len(h)$ do
        $input[i+j-1] -= h[j]$
      end for
    else
      $output[i] \Leftarrow 0$
    end if
  end for
```

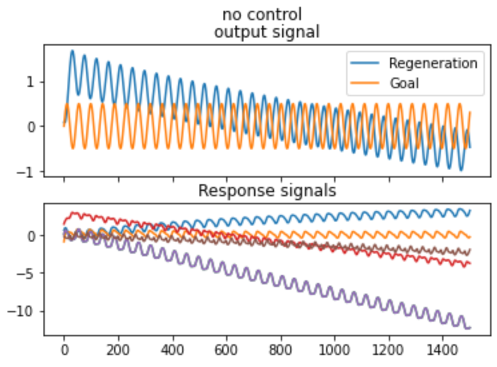
B Simulation Parameters

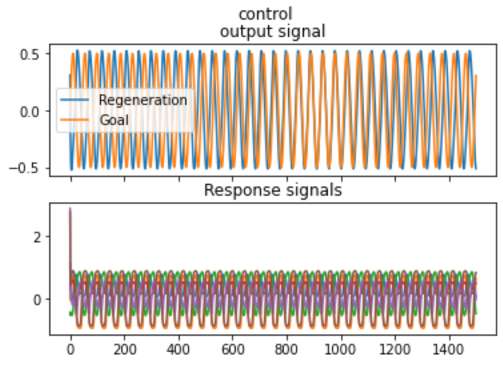
Reservoir Neuron parameters	
Spike threshold	15 mV
Reset potential	13.5 mV
Refractory period	3 ms
Time Constant	30 ms
Global Reservoir Parameters	
Reservoir size	10
Connection probability	50%
Scaling weight inhibitory	0.7
Scaling weight excitatory	0.5
Injection probability	100%
Miscellaneous Parameters	
Refractory period input	15 ms
Simulation length	5 s
BSA threshold	0.6
Sampling rate	10 Hz

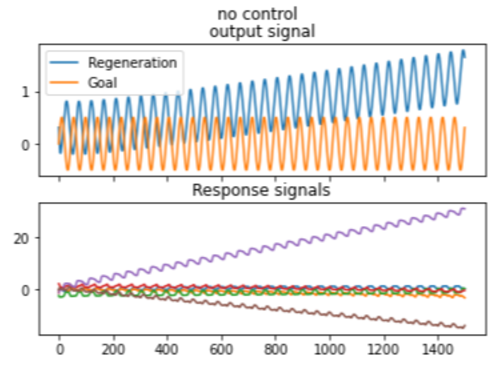
C Pattern regeneration output data

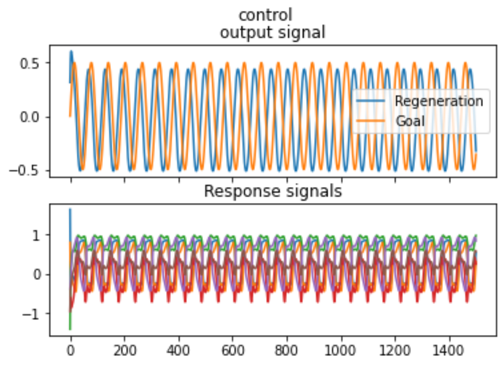
The goal of Experiment 1 was to investigate the quality of pattern regeneration using Equation (2.24). To do this, two conditions were compared, these conditions were defined by the value of the control constant K during retrieval. The first condition had a value of $K = 1.5$ this condition is called the control condition, the second condition had a value of $K = 0$ called the no-control condition. For each condition, the NRMSE between the retrieved signal (output using Equation (2.24)) and the goal signal (output obtained by driving the network on the input signal) was computed. These values provide a comparison between the control and no-control condition. However, inspecting the retrieved signals visually provides another manner to evaluate the quality of both conditions. The following plots show the data based on which the two NRMSE values reported in Section 4.1.1 were computed. Additionally, they serve to visually illustrate that having a non-zero value of K is necessary to obtain control/achieve pattern regeneration. The structure of the appendix is as follows; The left column shows data collected for the control condition, and the right column shows the data for the no-control condition. An element consists of two plots, the top shows the retrieval output and the goal output and the bottom plots neural response signals. Note that as per Section 3.1 a single trial consisted of four retrieval attempts, two signals were loaded into the reservoir and were attempted to be retrieved with control (2 runs) and without control (2 runs). This means that each page contains the results of two different independently created reservoirs. It is obvious that the control condition far outperforms the no-control condition. Besides the performance, the plots also give some insight into the reservoir behaviour for both conditions, as seen in the response signals.

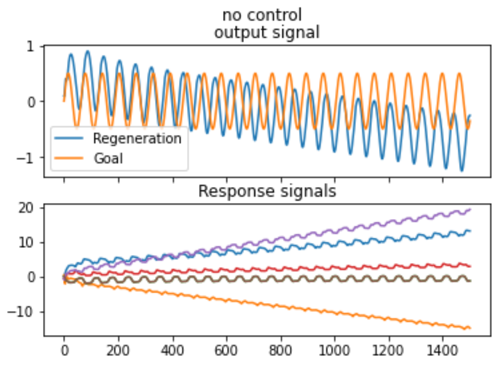

control
output signal
Regeneration
Goal
Response signals

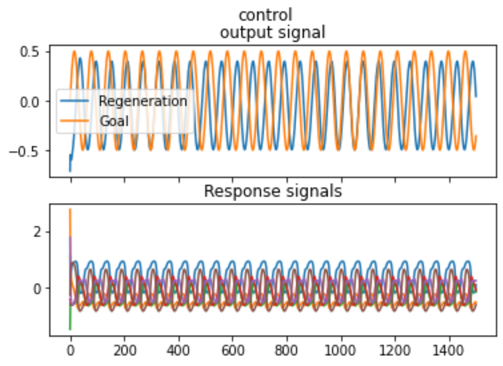

no control
output signal
Regeneration
Goal
Response signals

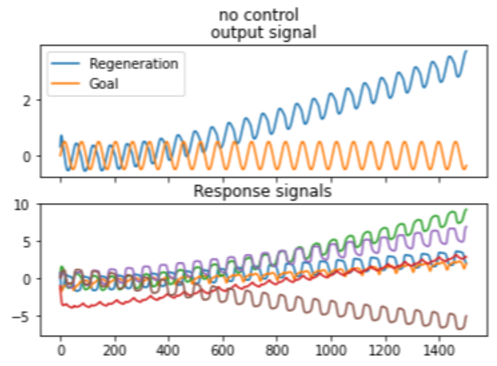

control
output signal
Regeneration
Goal
Response signals

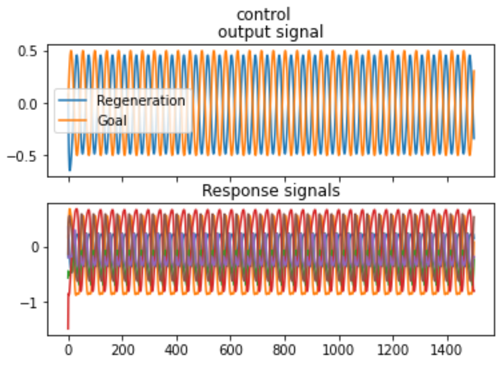

no control
output signal
Regeneration
Goal
Response signals

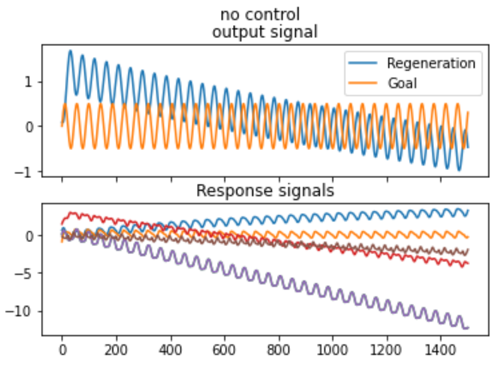

control
output signal
Regeneration
Goal
Response signals

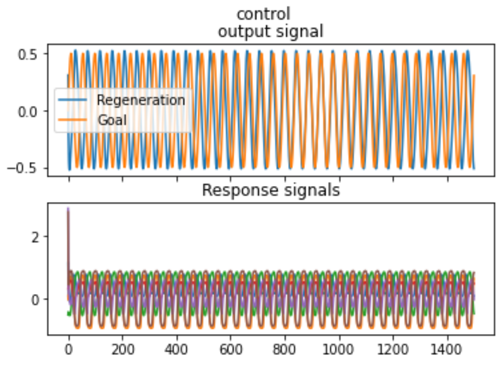

no control
output signal
Regeneration
Goal
Response signals

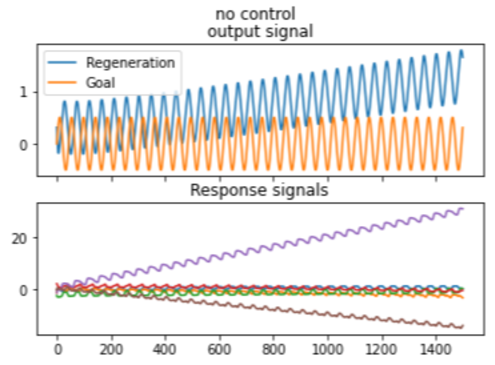

control
output signal
Regeneration
Goal
Response signals

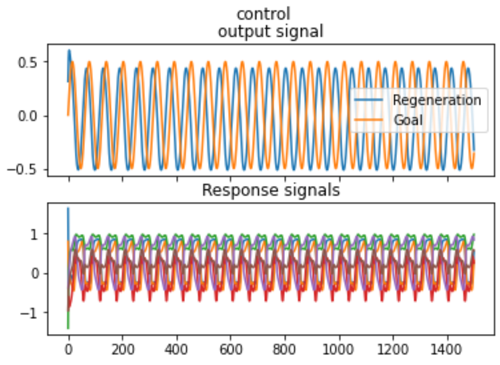

no control
output signal
Regeneration
Goal
Response signals

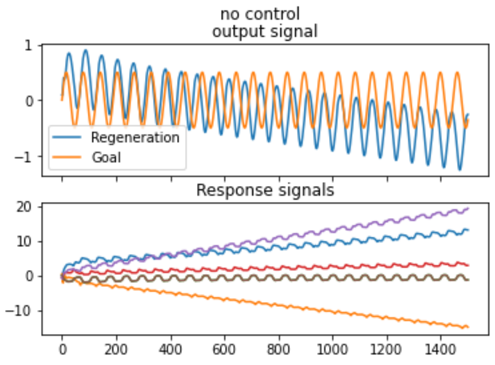

control
output signal
Regeneration
Goal
Response signals

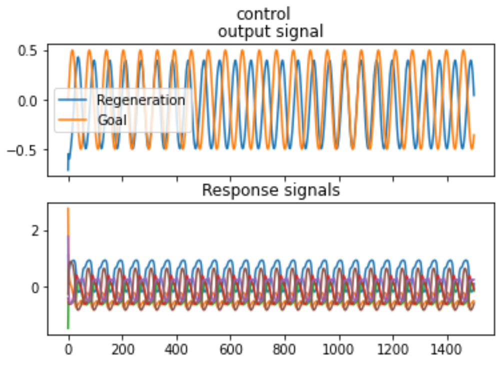

no control
output signal
Regeneration
Goal
Response signals

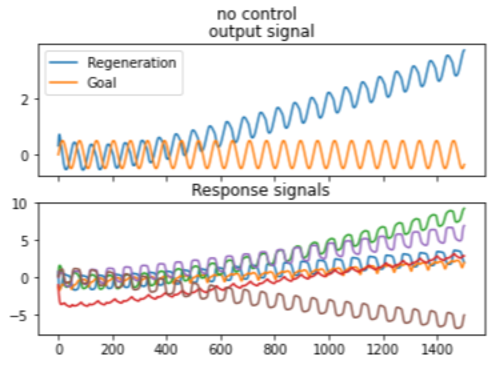

control
output signal
Regeneration
Goal
Response signals

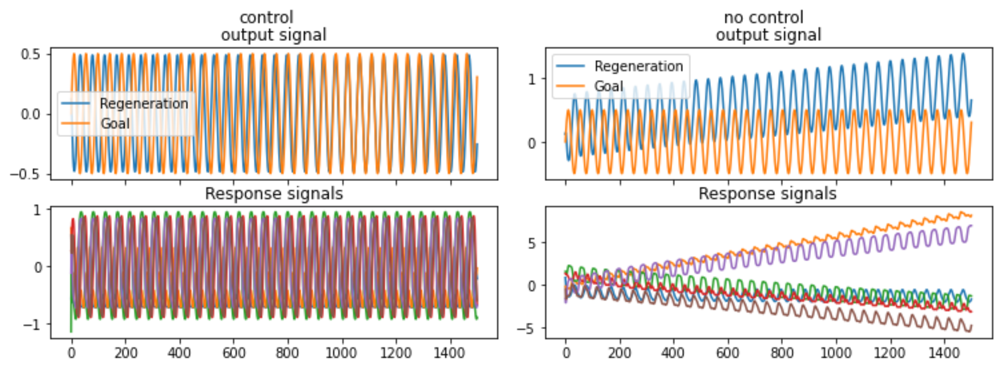

no control
output signal
Regeneration
Goal
Response signals

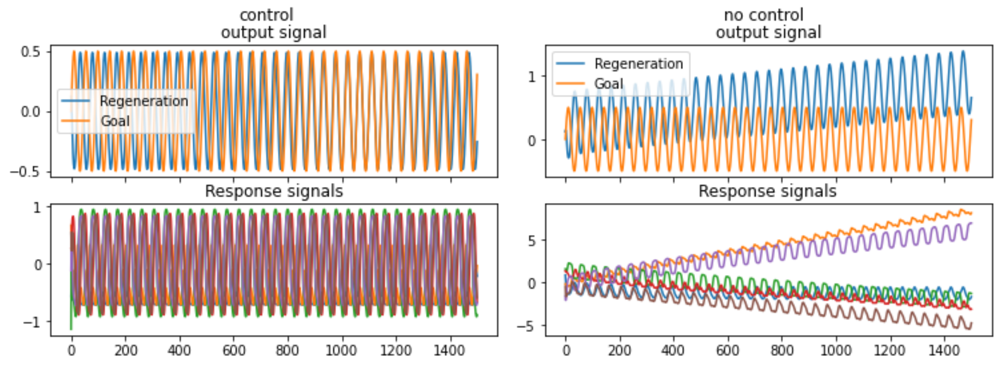

control
output signal
Regeneration
Goal
Response signals

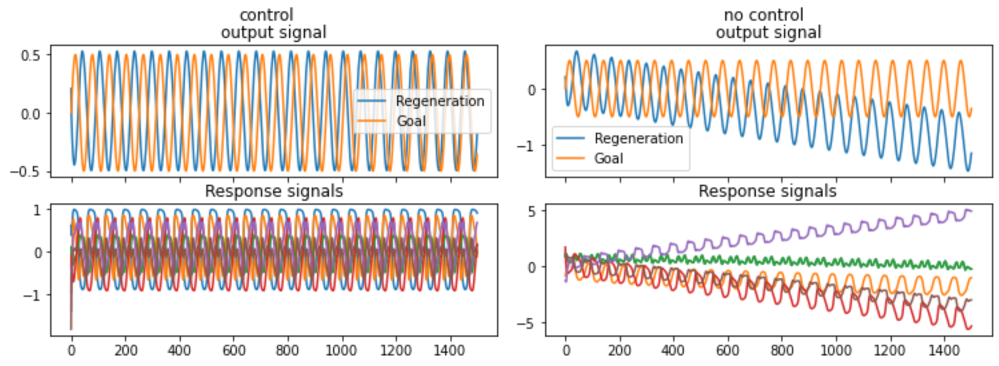

no control
output signal
Regeneration
Goal
Response signals

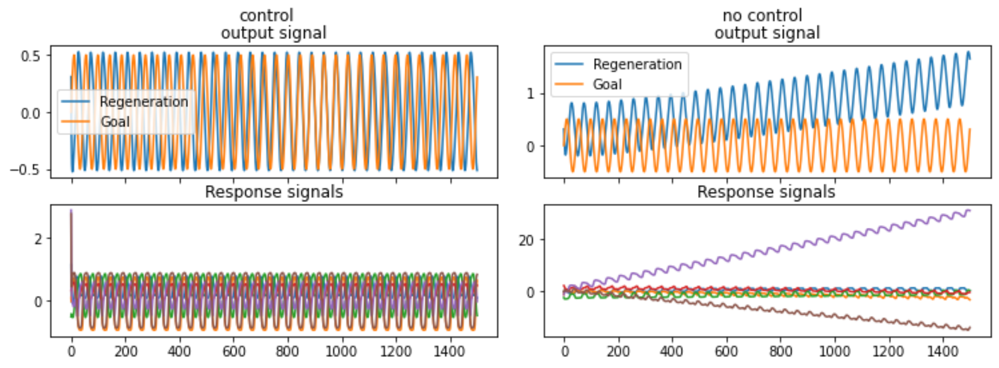

control
output signal
Regeneration
Goal
Response signals


no control
output signal
Regeneration
Goal
Response signals


control
output signal
Regeneration
Goal
Response signals
no control
output signal
Regeneration
Goal
Response signals


control
output signal
Regeneration
Goal
Response signals
no control
output signal
Regeneration
Goal
Response signals


control
output signal
Regeneration
Goal
Response signals
no control
output signal
Regeneration
Goal
Response signals


control
output signal
Regeneration
Goal
Response signals
no control
output signal
Regeneration
Goal
Response signals